coster group

PERSONAL AUTOMATION

WebGarage Modbus Integration

Modbus

We have created a new and improved workflow for integrating Modbus devices.

Prerequisite: Modbus Ext enabled in the Settings > Ext app.

Using Modbus

Once enabled, the user will go to the DB Builder app and expand the Connectors Tree to see the Modbus option.

Create Modbus Connector

Once Modbus is selected, the user would select the '+' and then "New Modbus" or Add button to create a new Modbus connector.

The user will be prompted with the above popup and the properties to fill out are:

- Dis: the name of the connector
- ModbusSlave: this would be the slave of the modbus device being connected (default is 1)
- Existing Register Map: If the user already configured a Modbus connector, they would see available registers here to choose from or create a new one with the next property ModbusRegMapUri.
- ModbusRegMapUri: this is where the user would specify the name of the register map to link with this connector. Replace "xxx" with whatever the name of the register map is.
- Uri: this is where the uri of the modbus connector would be specified. The user would enter the host, protocol and port if not standard default 502.
- **Supported protocols:** modbus-tcp (TCP/IP) or modbus-rtutcp (serial RTU over TCP)

Our example, we specified "test" as our register map. Then added our IP to the device in the Uri.

Once the connector is configured, the user would need to create a register map. (The register map could be created first and then chosen when creating the connector. No particular order needed here)

Create Register Map

To create or upload a register map, the user would select the "Register Maps" option just below Modbus option.

Then the user can either create a new one from scratch or upload one if they already have one as a CSV.

Where Register Maps Are Stored

Once a Register Map is created or uploaded, they get stored in the FIN Framework directory under the project > data > modbus folder. The path would look like something below. The user can create the folder manually and add the CSV files in there manually. They would need to stop and restart the service.

Modbus Folder Path: C:\Program Files (x86)\FIN Framework\FIN Framework 5.0.2198\var\proj\demo\data\modbus

Creating Register Map Manually

To create one from scratch, the user would select the "Add" button.

Once selected, the user will be prompted with the below pop up that will require them to name their register map.

In our case, when we created our connector, we specified "test" as being our register map so we'll name it "test".

Once the user creates the register map, it would become visible under the "Register Map" option where they can select it to edit it.

Once the register map is selected on the left tree under Register Map, the user can add more registers by selecting the Add button or manually editing it via the Edit Source option.

Note: Notice that a "ping" register is automatically added by default. This is required and needs to be a valid ping register in order to test connectivity to the slave. Modify it to match a register that exists or if you know what it is change it to that.

Uploading Register Map

If the user already has a register map with all their registers, they can upload one using the Upload button.

Once selected, the user will be prompted with the below pop up to find their CSV Register Map.

Once selected, the user will able to see the register map available in the grid view and on the left tree menu under "Register Maps".

From here, the user can then select the uploaded register map on the left side to edit registers if needed or add more.

Note: A "ping" register is required and needs to be a valid ping register in order to test connectivity to the slave. Modify it to match a register that exists or if you know what it is change it to that.

Deleting Register Map

If the user no longer needs a register map, they can select the register map after having selected the "Register Maps" on the left side. Then select the Delete button.

Adding New Registers

To add more registers, the user would select the "Add" button when selecting a register map on the left side menu.

The user will be prompted with the below pop up once selected.

The user can then configure the below fields:

- name (required) - name of register. It must be unique and must start with a lower case letter a-z and a-z,A-Z,0-9, and underscores are allowed. No spaces or other special characters.
- addr (required) - this would be the address of the register that follows the modbus convention.
- 0xxxx - Coil (00001-065536)
- 1xxxx - Discrete Input (10001-165536)
- 3xxxx - Input Register (30001-365536)
- 4xxxx - Holding Register (40001-465536)
- data (required) - this defines the data type of the register.
- Types:
- bit - Bool
- u1 - Unsigned 8-bit Int
- u2 - Unsigned 16-bit Int
- u4 - Unsigned 32-bit Int
- s1 - Signed 8-bit Int
- s2 - Signed 16-bit Int
- s4 - Signed 32-bit Int
- s8 - Signed 64-bit Int
- f4 - 32-bit Float
- f8 - 64-bit Float
- Bit Mask Types - supports a position notation for cases where bits are packed into input or holding registers:
- format - name, addr, data, rw
- do0, 40101, bit:0, rw
- do1, 40101, bit:1, rw
- do2, 40101, bit:2, rw
- Word and Byte Order - if register data not stored in network byte order, you can specify the order using suffix:
- u21e Unsigned 16-bit Int - Little endian byte and word order
- u21eb Unsigned 16-bit Int - Little endian byte order only
- u21ew Unsigned 16-bit Int - Little endian word order only
- rw (required) - this would determine the read/write permissions

- rw - Register may be read and written
- r - Register is read-only
- w - Register is write-only
- scale (optional) - this allows the user to apply a scale factor to the registers. The format is [operator] [number] where the factor is a numeric constant.
- Examples:
- add: +1.5
- minus: -0.25
- mult: *10
- div: /1000
- dis (optional) - this is an optional tag that the user can specify a pretty display name for the register
- unit (optional) - this would define the unit to use for the register
- tags (optional) - this would be the tags to apply to the point when learned into the database
- folderPath (optional) - if the user wants to organize the points, they can do so by applying a folderPath

Example of one of our points. We decided to include a unit and couple tags. The tags are "current" and "phase". The phase tag is a string with the value "A" in it.

Editing Registers

If the user needs to edit a register, they can select it from the grid view and then select the Edit button.

Once selected, they will be prompted with the below pop up to edit properties.

Deleting Registers

To delete a register from a Register Map, the user would select the register and then the Delete button.

Editing Register Maps via Edit Source

If the user wants to freely modify the register maps without the tools, they can do so using the Edit Source button. They can also copy from it and paste into a CSV file to modify there.

To do that, the user would first select the register map to edit on the left side menu under Register Maps and select Edit Source.

Once selected, the user will be prompted with the below popup to make any edits or copy to CSV.

Once changes are made, they can save them by selecting the Save button.

Viewing or Discovering Points

Once the user is done with their registers, they can then go back to the connector and select it to expand it to see the new register points or Discover Points to see them.

Adding Points to DB

From here, the user can now drag them out into their equips under Equip Tree to add points to db or use the Add button when in the correct context to add the points.

In our example below, we dragged the point from the Discover Points to the "Modbus Equip" we created.

Adding Points using Add Button

To use the Add button, first the user would need to be in the correct equip target context to add. To do this, the user can either navigate to the equip in the navigation tree or select "Make Current Context" button on the equip that would automatically switch the navigation tree to that target.

Once added to db, the user can see the tags and unit added if they were specified in the register creation.

Supported Protocols

Here are the supported protocols for Modbus. They are TCP/IP, RTU over TCP/IP, and RTU over RS-485.

The URI setup for each are as follows:

TCP/IP: `modbus-tcp://host/`

RTU over TCP/IP: `modbus-rtutcp://host/`

RTU over RS-485: `modbus-rtu://ttyUSB0`

Example Register Map Formats for Uploading

Below are the two CSV formats currently supported. Make sure the CSV files are saved as CSV UTF-8 if degree units are included.

History

History synchronization is not supported by Modbus. You will need to use our history collection to store history.